Automated Diagnosis of Breast Cancer Using Artificial Intelligence

Through the Implementation of Neural Networks
An Open Source, Cloud Implemented Diagnosis

Guss, William
william@wguss.com

Chen, Patrick
patrickbochen@gmail.com

March 16, 2014

Abstract

The challenge of diagnosing cancer is that no single test can accurately succeed. Diagnostic testing is essential to evaluate the health of an individual and determine whether the individual has cancer. Diagnostic imaging is a useful technique to produce an internal picture of the body for analyzing structure. However, medical professionals are required to successful analyze the images and determine whether the individual has cancer. Applying artificial neural networks to this problem makes the analysis more efficient while minimizing error in diagnosis.

The purpose of the project is to implement a successful neural network with backpropagation to analyze a breast cancer numerical and image dataset. It also evaluates the efficiency of the network as it is influenced by different conditions. The efficiency is gauged by the error percentages accumulated by the network. Furthermore, statistical analysis is applied to the network in order to analyze the effectiveness.

The project showed that despite the adaptability of the neural network, it is still unable to remove the error completely. While neural networks are useful, they cannot be relied on completely. However, there is a tradeoff between error and flexibility in the network. While the error has the potential to be removed from the testing of the neural network, the network would be over fitted to the dataset it is being trained and tested on so that it would lose its ability to successfully analyze similar forms of data.

Contents

1 Introduction

Breast cancer has become one of the most common existing cancers. It is a frequent and leading cause of mortality, especially in developed countries. This risk of receiving the cancer also increases with age. However, early detection of breast cancer leads to increases in survival rates. Moreover, doctors may employ several methods to determine the existence of cancer. One diagnostic procedure that investigates potential lumps in the breast is called a fine-needle aspiration biopsy (FNA). These biopsies are very safe and minor surgical procedures making them viable options for wide usage for diagnosis. A thin hollow needle is inserted into the breast to sample cells, which are then stained and examined under a microscope. A pathologist or another expert that examines the biopsy will then determine the state of the breast based on observations made from the data. However, due to the fact that the final decision is subjectively made by the pathologist, there is still large room for human error. Thus implementing artificial intelligence, specifically neural networks, is a more accurate and efficient way of processing the data and determining the malignancy of the breast.

Despite these options, the most current diagnostic method for early detection of breast cancer is a mammography. These are x-rays of the breasts that detect micro-calcifications. Micro-calcifications are tiny bits of calcium that usually indicate extra cell activity in the breast tissue which can indicate early breast cancer. However the majority of scattered micro-calcifications, which appear as white speckles on a mammography, are benign. There micro-calcifications range from one hundred micrometers to two millimeters. Because of the difficulty in locating and analyzing these aspects and because survival from breast cancer is dependent on early detection, a computer aided diagnosis is useful and beneficial to detect micro-calcification clusters.

Because, these micro-calcifications represent such a large range of possible image datasets, a complex function must be used to model the existence of these in relation to the benign or malignant state of the cells in the breast. A neural network is more optimal for determining this function as opposed to normal algorithmic approaches that are unable to model the function as well. Because increasing the survival rate from breast cancer is dependent on correctly identifying the cancer to begin with, using a more effective method with low error rates such as a neural network is preferred.

1.1 Goals

- Completely understand and implement neural networks.
- Understand the math and logic behind backpropagation and implement the algorithm correctly.
- Demonstrate the flexibility of neural networks.
- Train the neural network on a dataset and test the effectiveness of the neural network.
- See the relationship between different variables in the dataset.
- See the effect of variables in the network algorithm on the error produced.
- Statistically analyze the data produced by the neural network.

- Demonstrate the effect of weights on the network and the randomness of the weights that lead to different solutions.
- Solidify programming abilities and gain a foothold in programming for artificial intelligence.
- Implement image recognition through the use of artificial intelligence and employ an image dataset to demonstrate the versatility of neural networks.
- Create an application and cloud service that can be used in the medical field to alleviate the workload while diagnosing breast cancer.

1.2 Expected Outcomes

- The neural network is expected to work properly. Even though the network has a margin of error, that margin is what also allows the network to be flexible and correctly diagnose cases outside of the training dataset.
- The network is able to self organize itself and have a low percentage error for identifying malignancy in the input data.

1.3 Hypothesis

[Using multiple neural networks, accurate diagnosis of breast cancer malignancy in different mediums (numerical and visual) can be achieve with a low margin of error.]

2 Background

2.1 Neural Networks Compared to Conventional Approaches

Neural networks take a different approach to problem solving than conventional algorithmic approaches. Algorithmic problem solving requires a fixed set of actions to determine a solution, and if absent, an algorithmic function for such a problem is impossible, restricting the problem solving capabilities of conventional computing. Neural networks, in contrast, learn by example and cannot be programmed to perform a specific task, allowing a computer system to approximate an otherwise unknown function. Neural networks, as a result, are able to perform adaptive learning, self-organization, real time operation, as well as fault tolerance. The reliability of a neural network fails to match that of a conventional algorithm, as operation under certain conditions can be unpredictable.

Often, to combat uncertainties in the neural network and the limitations of an algorithmic approach, the two problem solving methods are combined to perform a task at high efficiency.

2.2 Similarities Between Neural Networks and Humans

Neural networks are a form of artificial intelligence that derives from the structure of biological nervous systems and data processing methods within those systems. Both the brain and neural networks are composed of a large number of processing elements, referred to as neurons, which are interconnected by weights (in artificial neural networks) or axons and dendrites (in biological neural networks), which work together with the neurons to solve problems. Artificial networks and an organism's nervous system learn by example. These two structures update their weights or connections in response to inputs and whatever result is desired.

Whereas artificial networks are typically data intensive and thus limited to several hundred units, biological neurons can consist of 10,000 individual inputs, immensely more complex. The complexity of the existing neural network is limited by the computing power of the computers or artificial systems in place today.

2.3 Neural Networks

To account for the lack of an algorithm, neural networks attempt to discern patterns from a dataset. Although any dataset is able to work, a larger dataset is typically required to allow a more adaptive nature for the neural network. The neural network takes in a set of inputs and passes on the values through a series of numbers termed weights, which then pass on an altered value to neurons in subsequent layers (either hidden or output). Once the network outputs values, the results are compared to the desired output, and the network's weights are adjusted accordingly, through the method of backpropagation.

The feed forward neural network is the simplest type of an artificial neural network utilized. Information moves in a single direction, forward, from the input nodes, through hidden nodes (if any) and to output nodes. There are no cycles or loops in the network. Feed forward networks are the most popular and widely used function-modeling structures that reflect a dataset.

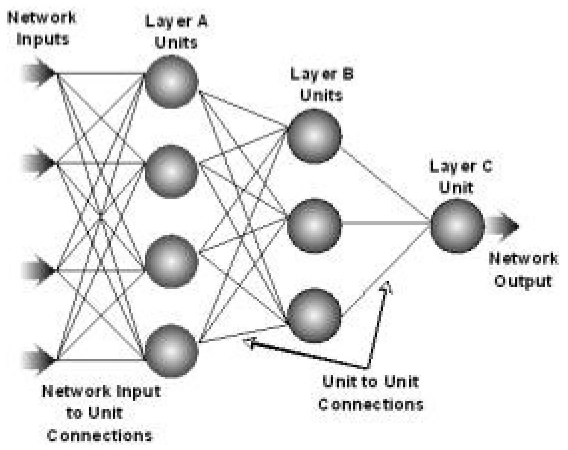

Figure 1: A diagram of a feed-forward artificial neural network.

2.3.1 Advantages

1. Adaptive learning: An ability to learn how to do tasks based on the data given for training or initial experience.

2. Self-Organization: A neural network can create its own organization or representation of the information it receives during learning time.

3. Real Time Operation: Neural network computations may be carried out in parallel, and special hardware devices are being designed and manufactured which take advantage of this capability.

4. Fault Tolerance via Redundant Information Coding: Partial destruction of a network leads to the corresponding degradation of performance. However, some network capabilities may be retained even with major network damage

2.3.2 Limitations

1. Limited by the computing power of existing computers or current artificial systems.

2. Unpredictable at times

3. Randomness due to the initial starting weight values which are determined and randomly based off a Gaussian distribution curve. At times, the randomness leads to the inability of the network to converge.

4. Unable to completely remove error because of the necessity of some in order to retain the adaptive nature of the network.

5. Extremely hard to optimize and minimize the error because of the numerous amount of variables that make up the network and affect it.

2.4 Error Backpropagation

Backpropagation, or backward propagation of error, is the most common algorithm for training neural networks. It is used to find a function that best models the input data. The goal of the backpropagation algorithm is to train any neural network such that it can learn to create any arbitrary map from inputs to outputs. From the error between the desired output and the actual output, the network readjusts its weights to reduce the error being produced. Backpropagation is most useful for feed forward networks which have no feedback.

2.4.1 Intuition

Each neuron employs a linear output that is the weighted sum of its input. Initially, before training, the weights will be set randomly. Afterwards, the neuron learns from training input by employing the sum squared error to measure the discrepancy between the expected output and the actual output. Therefore, the problem of mapping inputs to outputs can be reduced to an optimization problem of finding a function that will produce the minimal error. The backpropagation algorithm aims to find the set of weights that minimizes the error. The method used in backpropagation, to find the minima of a function in any dimension that models that map from input to output which minimizes the error, is gradient descent.

2.4.2 Derivation

Error backpropagation is derived through a series of chain rule calculations based on the sum squared error function of the neural network. The error function is composed of a series of output vectors which themselves are composed of weight vectors and output vectors from activation functions of neurons from previous layers. The essential goal of this derivation is to find a suitable weight update rule such that the error function of the neural network descends to a local minimum using a weight gradient vector.

Take the sum squared error function,

$$E = \frac{1}{2}\sum_{j\in L}(d_j - \sigma_j)^2$$

where L is the output layer and j is a given node on the output layer. To calculate $\frac{\partial E}{\partial w_{ij}^L}$, chain rule is direct given a lack of composite pathways to that weight. Therefore the partial derivative is given by

$$\frac{\partial E}{\partial w_{ij}^L} = \frac{\partial E}{\partial \sigma_j^L}\frac{\partial \sigma_j^L}{\partial \mathrm{net}^L}\frac{\partial \mathrm{net}^L}{\partial w_{ij}^L}$$

Given that $\sigma = \tanh(\mathrm{net})$ and $\mathrm{net} = \sum w_{ij}\sigma_i$, we calculate the (i, j) component of the weight gradient on the output layer to be

$$\frac{\partial E}{\partial w_{ij}^L} = (d_j - \sigma_j^L)\sigma_j^L(1 - \sigma_j^L)\sigma_i^{L-1}$$

Although this calculation of weight gradient is applicable to the last layer of weights, a generalization must be made using higher-order chain rule to determine the weight gradient for a weight on any given layer, l. For the sake of simplicity we will define a node delta for the output layer as

$$\delta_j^L = \frac{\partial E}{\partial \sigma_j^L} \frac{\partial \sigma_j^L}{\partial \text{net}^L}$$

Thereafter follow that each weight gradient is directly related to every pather leading to the weight, itself. Therefore for the hidden layer, $l = L - 1$, we must define the node delta as a summation given by

$$\delta_i^{L-1} = \sum_{j \in L} \frac{\partial E}{\partial \sigma_j} \frac{\partial \sigma_j}{\partial \text{net}_j} \frac{\partial \text{net}_j}{\partial \sigma_i} \frac{\partial \sigma_i}{\partial \text{net}_i}$$

Using the gneralization employed in deriving the node delta for a given neuraon in the first hidden layer, we can then state the generalization for any hidden layer to be

$$\delta_i^l = \sum_{j \in (l+1)} \delta_j w_{ij} \frac{\partial \sigma_i}{\partial \text{net}_i}$$

Now that we have a model for calculating the node delta for any given neuron in any given layer, let

$$\frac{\partial E}{\partial w_{ij}^l} = \delta_j^{l+1} \sigma_i^l$$

To calculate the change in weight, error backpropagation then uses the method of gradient descent to define the weight update as

$$\Delta w_{ij}^l(t) = -\lambda \frac{\partial E}{\partial w_{ij}^l} + \mu \Delta w_{ij}^l(t-1)$$

where λ is a user-defined learning rate and μ is a momentum term which avoids local minima trapping.

2.5 Haar Wavelet Transform

Haar wavelet is a subset of a larger concept known as wavelet transform. Wavelet transformations are meant to change the time-frequency of an image. The transformation however only occurs in the time extension while maintaining the shape. An application of wavelet transformation is to compress images while reducing loss of data and is used instead of the Fourier transformation because it is able to capture both frequency and location information. Inside wavelet transforms, there are two sub-categories, discrete and continuous. Discrete is represented by integers while continuous can be represented over an entire range of numbers. Discrete is preferable for image recognition because it is used to represent pixel values which are whole numbers. These pixel values once altered by wavelet transform are then used as input for a neural network.

The exact wavelet transform used during this experiment was Haar wavelet transform. It is a transform that creates a sequence of rescaled rectangular shaped functions. For the purpose of this experiment, the transform is meant to reduce the data that is being inputted into the neural network to increase processing speed. It does so by using minimal pixels and using multiple scales which are represented simultaneously. Other advantages of the process are that it is both invertible and linear.

2.5.1 Decomposition

Given an original image with pixel values P_1,P_2,P_3, and P_4 as represented in diagram below and a size length of 2^n where n is an arbitrary integer.

P_1	P_2	...	...
P_3	P_4	...	...
...	...	...	...
...	...	...	...

After one wavelet transform, the pixel values of the new image are represented by

$(P_1 + P_2 + P_3 + P_4)/4$	...	$(P_1 - P_2 + P_3 - P_4)/4$	...
...	...	...	...
$(P_1 + P_2 - P_3 - P_4)/4$	...	$(P_1 - P_2 - P_3 + P_4)/4$	...
...	...	...	...

The two basic operations that occur are the sum of the four pixels and the difference of pair wise sums of pixels to create a new image. This process continues for part of the image, specifically for the length range of 0 to 2^{n-1}. This process will continue recursively until the length range reaches two. It is meant to reduce the amount of input to the neural network while keeping the amount of information constant.

3 Datasets

3.1 Breast Cancer Wisconsin (Original) Dataset P-FNA Test

This dataset is numerical and multivariate with integer attribute values. The creator was Dr. William H. Wolberg at the University of Wisconsin in Madison, Wisconsin. The diagnostic test created for this specific dataset is called a P-FNA test, proportional fine needle aspiration test.

The dataset used had 9 input attributes, each from a range of 1 to 10. There were a total of 699 data points. However, 16 of the points had inconsistencies where a question mark stood in place of a number. The 16 data points thus were excluded from both the training and the testing of the neural network. 10% of the data was used for testing while the other 90% was used for training.

While the data had the output of 2 for benign and 4 for malignant, during the testing of the network these numbers were changed to 0 and 1 for benign and malignant respectively. This is because the logistic function can only output from a range of -1 to 1. For the actual dataset, 65.5% were benign and the other 35.5% were malignant. The source also claimed that there is also a 5% discrepancy in the dataset.

3.1.1 Inputs

Clump Thickness
: Benign cells tend to be group in a monolayer, while cancerous cells are often grouped in a multilayer.

Uniformity of Cell Size
: Cancer cells tend to vary drastically in size and shape, thus a lower uniformity correlates with a higher possibility of cancer cells.

Uniformity of Cell Shape
: Cancer cells tend to vary drastically in size and shape, thus a lower uniformity correlates with a higher possibility of cancer cells.

Marginal Adhesion
: Cancer cells tend not to stick to one another as well as normal cells, so less adhesion correlates to a higher malignancy.

Single Epithelial Cell Size
: The size is related to uniformity, but enlarged epithelial cells may be malignant.

Bare Nuclei
: It is an index of nuclei not surrounded by a cell, which is present in malignant tumors.

Bland Chromatin
: Uniformity of texture appears in benign cells, while malignant tumors are typically coarse-textured. A lower number corresponds to more unity.

Normal Nucleoli
: The rate of occurrence of normal nucleoli; abnormal nucleoli indicate possible mutated DNA, thus possible genetic expression for cancer reproduction. Thus, the smaller the rate of occurrence, the larger the chance of malignancy becomes.

Mitoses

Cancer cells tend to replicate faster which contributes to a tumor and leads to increased potential in harmful consequences. Thus, a set of cells with a higher rate of mitoses has an increased chance of being malignant.

3.2 Breast Cancer Wisconsin (Diagnostic) Dataset D-FNA Test

This dataset is numerical and multivariate with real attribute values. The creator was Dr. William H. Wolberg at the University of Wisconsin in Madison, Wisconsin. The diagnostic test created for this specific dataset is called a D-FNA test, detailed fine needle aspiration test.

The dataset used had 30 input attributes, each represented by a real value with four significant digits. Ten features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image. The mean, standard error, and largest of these features were computed for each image, resulting in 30 real valued features. There were a total of 569 data points. 10% of the data was used for testing while the other 90%t were used for training.

While the data had the output of B for benign and M for malignant, during the testing of the network these numbers were changed to 0 and 1 for benign and malignant respectively. This is because output of the neural network is a numerical value. For the actual dataset, 62.7% were benign and the other 37.3% were malignant.

3.2.1 Inputs

Radius

The mean of the distances from the center of the nucleus to points on the perimeter.

Texture

The standard deviation of grey-scale values of the FNA.

Perimeter

The perimeter of the nucleus.

Area

The area of the nucleus.

Smoothness

The local variation in radius lengths.

Compactness

Calculated by $\frac{P^2}{A} - 1$, where P is the perimeter and A is the area.

Concavity

The severity of concave portions on the countour of the nucleus.

Concave points

The number of concave points on the contour.

Symmetry

An average measure of symmetry across the nucleus.

Fractal dimension
The coastline approximation less one is the fractal dimension, which is also noted to be an objective and reproducible measure of the complexity of the tissue architecture of the biopsy specimen.

3.3 Mammographic Image Analysis Society (MIAS) Database MID Test

This dataset is composed of images from mammographies. The creator was the Mammographic Image Analysis Society. The diagnostic test created for this specific dataset is called a MID test, mammography image diagnostic test.

The dataset is composed of images with 200 micron pixel edges. Each image size is 1024x1024 pixels. The dataset used for input to the neural network is called a mini-MIAS database created by J. Suckling who reduced the resolution of the original MIAS database. There were a total of 322 data images. 18.2% of the data was used for testing while the other 81.8% were used for training. However, only a small portion of the actual images were used. The breasts that were in the dataset had three distinct background tissues: fatty, fatty-glandular, and dense-glandular. Only images with fatty background tissue were used during the experiment to reduce a variable for the network. Also only images of left breasts were used to reduce the amount of uncertainty in the network because of the existence of black space on either side of the breast. Once these specific images were separated from the original dataset, a total of 22 images remained. Of these 11 were tumorous and 11 were normal (more normal ones existed but in order to achieve consistency during training and testing only 11 were used).

While the data had the output of N for normal and M for malignant, during the testing of the network these numbers were changed to 0 and 1 for normal and malignant respectively. This is because output of the neural network is a numerical value. For the actual dataset, 50% were malignant, and the other 50% were normal.

4 Implementation

4.1 Nudging

Nudging was implemented into the network. The point of nudging is to ensure that the network does not get trapped in a local minimum when training. At times because of the randomly distributed starting weights, the training of the network using backpropagation would become trapped in a local minimum that does not reach the desired error for convergence rate. If this occurs to the network, then nudging needs to occur in order to converge at the predetermined minimum error value. Reducing the error of the network is extremely important for diagnosing breast cancer because the misdiagnoses of the disease are life threatening.

Usually once the difference between ten epochs is less than .0001, then the weights would randomly adjust based on a Gaussian distribution. However, in the neural network created, nudging was based on standard deviation instead of difference. The network would then train again using these new weights. The rationale is that once the change becomes so small, then the network has become stuck in a local minimum without the ability to escape and also has not converged yet.

4.2 Acclerative Learning

During the course of the project, it was realized that a more efficient and accurate algorithm could be developed to counteract the effects of local minima trapping in the error-backpropagation algorithm. Although nudging provides a brute-force solution to the aforementioned problem, much data loss occurs due to the re-initialization of weights throughout the entire network.

We thereafter developed an algorithm stemming from both standardized error backpropagation and resilient error backpropagation. The algorithm follows the same gradient derivation as error backpropagation but updates weights differently; instead of using a direct proportionality rule to change weights, the accelerative learning rate algorithm employs both a momentum and an acceleration based on the continued homogeneity of sign from previous iterations.

The proposed algorithm is derived as follows. Firstly, we calculate the gradient of a given weight using the partial derivative calculated through chain rule given by

$$\frac{\partial E}{\partial w_{ij}^{l}} = \delta_j^{l+1}\sigma_i^{l}$$

Similar to resilient propagation, we then consider a change in sign based on the product of the current weight gradient and the weight gradient from the previous iteration. If the product is positive then the sign of the gradient has remained homogenous, whereas if the product is negative the gradient has changed in both sign and direction. In the case that the product results in zero, either the gradient from the previous iteration or that from the current iteration is stationary. From the product, we then determine a change in velocity given by

$$v(t) = \begin{cases} v + \alpha & \text{if } [\frac{\partial E}{\partial w_{ij}^{l}}(t) \cdot \frac{\partial E}{\partial w_{ij}^{l}}(t-1)] > 0 \\ \alpha & \text{if } [\frac{\partial E}{\partial w_{ij}^{l}}(t) \cdot \frac{\partial E}{\partial w_{ij}^{l}}(t-1)] = 0 \\ 0 & \text{if } [\frac{\partial E}{\partial w_{ij}^{l}}(t) \cdot \frac{\partial E}{\partial w_{ij}^{l}}(t-1)] < 0 \end{cases}$$

Thereafter, the weight is updated according to an accelerative proportionality to the gradient. The algorithm here differs from resilient error backpropagation in that not only is proportionality maintained with the gradient but also a continued momentum is employed likened to that of modified backpropagation algorithms. The weight update rule is then given by

$$\Delta w_{ij}^{l}(t) = -v\frac{\partial E}{\partial w_{ij}^{l}} + \mu \Delta w_{ij}^{l}(t-1)$$

An advantage to the accelerative learning rate algorithm is a given control over learning parameters. As seen in Reidmiller (1998), the error backpropagation algorithm fails frequently due to local minima trapping. Therefore Reidmiller proposed the resilient propagation algorithm. The RPROP algorithm uses a velocity update rule that disregards the magnitude of the gradient at a given point in weight space. Although highly adaptive, we found that in our implementation of both the RPROP and BPROP algorithms, there were still scenarios in which the weight update was either too sensitive or too broad. The accelerative learning rate algorithm solves this through employing an adaptive weight update which remains proportional to the gradient at a given position in weight space. The aforementioned approach allowed us to finely tune the convergence of the network whilst also avoiding local minima trapping.

4.3 Data Processing

The training dataset of the original Wisconsin Dataset was altered first by randomly excluding 68 data points, or 10% percent, of the 683 original dataset. 10 different neural networks were created by training them on the training dataset and their weights were then saved. These sets of weights are radically different from each other because of the existence of local minima, random weight space, and a preset convergence at .004% error. The 68 data points excluded were used as the testing data points. These data points were independently ran through the networks and then the probability of malignancy was recorded. A step function was then implemented to heavily weight the results of the network towards the malignant output. If the output was greater than .05 then the network would automatically consider the output for that data point to be malignant. Then, the output of the network was compared to the desired actual output. The average error for a single specific data point over all 10 networks next. Furthermore the average error for the total network is calculated by the average of the error for each data point. These calculations result in the total error of the network given by

$$E = \frac{1}{T}\sum_{x\in T}\frac{1}{N}\sum_{n\in N}\frac{1}{2}\sum_{o}(d_o - \operatorname{stp}(\sigma_o))^2$$

The same process occurred for the diagnostic Wisconsin Dataset, where 59 data points of also 10% were excluded for training. However, a difference that occurred is that the training would continue until the testing dataset reached an error of 2%, which is then considered the total error over all ten networks. The final or most optimal step function point was at 0.3. Because of the life threatening issue diagnosing breast cancer, the output is more weighted towards the malignant side not only to reduce error but to reduce the probability of a type 2 statistical error occurring. Further experiments occurred on this dataset to determine the best network variables, such as momentum and accelerative learning rate, for this network and specific dataset.

4.4 Network Structure

The networks created for each dataset all had different structures because the different datasets had different numbers of input values. For the original Breast Cancer Wisconsin dataset there were nine input neurons, two hidden layers were created of sizes 16 and 6 neurons in that order, and finally one output neuron was created. The justification for these sizes are that the number or weights should be less than or approximately the number of datapoints in the dataset (Lee Cun et al.). Thus, for the second dataset, diagnostic Wisconsin Breast Cancer, there were 30 input neurons, 2 hidden layers of sizes 25 and 16 respective to their layer position, and one output neuron. This justification however does not apply to images. Each image should be considered an input neuron rather than its pixel values because during images in neural networks employ weight sharing. The network structure then for the image dataset is 64 inputs that represent the pixel values of the input image, 10 neurons in the second layer, and 5 neurons in the third layer, where both the second and third layers are hidden layers, and finally one neuron in the output layer.

4.5 Experiments (D-FNA)

For all the experiment and other various data processes, the initial weights of 10 different networks that would converge according to the control experiment were recorded. Then these initial weights were loaded into the other experiments as starting points for the networks. This is meant to keep variables constant, especially due to the random nature of the starting weights when normally initializing a neural network.

4.5.1 Control Experiment

The control experiment was the initial experiment. It just trained the network on randomly chosen values for the network size, momentum, and accelerative learning rate to determine the initial error of the network, which was the basis for the rest of the networks. Ten networks were trained and successfully converged at a maximum of 5% error. However, the max error was less than the preset amount because each iteration of error backpropagation changes the network by an unknown amount according to each networks position in weight space.

4.5.2 Step Experiment

The step experiment cycled through various step values from 0-1. It started over a wide range but after a single experiment the step size that created the lowest error was located and then another step experiment was performed on a smaller range centered at the previously located step size value. The best step size value located was 0.86 for the original test. However, after all the other variables were optimized during the conclusion experiments, it was demonstrated that the best step size became 0.3. This shows that the variables defining the network are extremely codependent on one another. Moreover, the change in optimum step-size also depicts an increase in accuracy by further weighting of false-positives over false-negatives. The trend for the malignancy weighting was parabolic; indicating that the error is becomes smaller as the weighting is centered on about 0.5. Thus, extreme weighting towards either side of the range of step values is actually detrimental to the error produced by the network.

4.5.3 Accelerative Learning Rate Experiment

The accelerative learning rate cycled through various accelerative learning rates in a similar fashion to the step experiment. However there were multiple experiments done, in which the range was decreased as the change was also reduced. In this instance all, the other variables such as momentum and network size were kept constant. The best accelerative learning rate for reducing the network diagnosis error was determined to be 0.025 from the range of $[0, 2]$, 0.000625 from the range of $[0, 0.025]$, and 0.00034375 from the range of $[0, 0.000625]$. The graph produced had many local minima and maxima. Overall though, there is a general increasing trend for the learning rate. As the rate increases the error produced by the networks would increase. However, the graph for this experiment was extremely varied and random, leading to no absolute relationship being able to be defined for accelerative learning rate and error.

4.5.4 Momentum Experiment

The momentum experiment was performed in a similar manner to the accelerative learning experiment. The independent variable in the case of the momentum experiment was the momentum coefficient in the weight update rule. The ten networks from the control experiment were retrained using the the variable change in momentum for each trial. The experiment located the best momentum value of the Neural Network at 0.15. The general trend for the network momentum is increasing. As the momentum approaches 1.0 the error will also increase. However there were many local minima and maxima that reflected a near sinusoidal pattern.

4.5.5 ALR-Momentum Covariance Experiment

During the covariance experiment, both the accelerative learning rate and momentum were altered to identify the best values for those two variables. This was done by keeping the accelerative learning rate constant for one trial while testing the momentum, then changing the accelerative learning rate by a set amount and running it through a momentum test again. The best values for accelerative learning rate and momentum identified are then 0.000171875 and 0.25 from the range of $[0, 0.000625]$ and $[0, 1]$ for accelerative learning rate and momentum respectively. This highlights the fact that accelerative learning rate, momentum, and codependent on altering the error rate of the network. In fact, an even lower error was achieved during the covariance experiments that had final optimal values vastly different from that of either the accelerative learning rate of momentum experiments. The data for the 3-D graph of error by covariance of accelerative learning rate and momentum indicated that there was a definite minimum. As the both the accelerative learning rate and momentum approach zero there is a definite trend of the error decreasing. However, once the learning rate reaches zero, there is an instantaneous significant increase in the error produced by the network. In addition, the absolute minimum did not reach zero error indicating that the learning rate and momentum are dependent on the initial starting weights for the error produced.

4.5.6 Conclusion Experiment

For the conclusion experiment, the size of the neural network was increased. This is because during the other experiments, it was realized that with the current network size the desired maximum error could not be reached. Thus the network size was increased to decrease the current error. In addition, the rest of the results from the previous experiments were applied to the network. The optimal accelerative learning rate and momentum identified during the covariance experiment were employed. Also, the 10 network structure with constant started weights used in the previous experiments was not used. Instead networks were created with random starting weights and trained until they reached the threshold error of 2%, which is the desired error, and the final error of the network. This done is to indicate the usefulness of the network at this time and how it operates under random starting weights with these predetermined and optimal variables that affect the network. The experiment demonstrates that despite random starting weights, the networks are able to converge due to the optimal values found for the variables experimented on. This finally leads to the overall goal, to demonstrate the flexibility and self-structuring nature of the neural networks. In addition, by comparing the output of the conclusion experiment to the output of the control experiment, it is demonstrated that the optimal values located by the previous experimented lowered the error produced by the network. These parameters are extremely important for implementing and optimizing a neural network.

4.5.7 Collection of Data

For each datapoint in each dataset the initial and ending weights were recorded along with the error per epoch of training for every network. In addition all the data for during the data processing was recorded. Over 20 million numbers were recorded for the entirety of the experiment. In the interest of conserving paper the data was not printed. However, to give a tangible indication of the amount of data collected, over 1.2 gigabytes of data were collected.

4.6 Images (MID)

The success of the MID Test was dependent on propper preprocessing and transformation of the images into feature vectors that could propigate through the network in reasonable time. These steps were accomoplished by employing the discrete wavelet transform and a conversion of pixel values to numerical values for input into the neural networks.

4.6.1 Preprocessing

The images received from the mini-MIAS database had to be preprocessed. The images were in the format of pgm and were converted into png using GIMP. Many of the images had differing amounts of black/blank space in them. Thus, only the actual area of interest, the breast, was cropped from the image using Photoshop into the resolution of 512x512 pixels. The specific number of 512 was employed because it is optimal for Haar Wavelet transform as it is a value of 2n where n is equal to nine in this instance.

4.6.2 Discrete Wavelet Transform

The Haar Wavelet transform was then employed on the processed images. A specific section was then selected for the use as input for the neural network. The Haar Wavelet program created three files. One was a png file of the image after Haar Wavelet was performed, another was a png file of the area of interest that was being inputted into the network, and the third file was a text file that held the pixel values of the area of interest. The area of interest was an 8x8 area where the Haar Wavelet transform had reduced the size of the image and consequently the amount of input to the neural network. Of the regions LH (intensity varies along the columns), HL (intensity varies along the rows), and HH (intensity varies along the diagonals), HH region was chosen for the area of interest.

4.6.3 Training

The text file of the previous was then run through 10 neural networks with 64 pixel values for each data point. The networks were trained on 18 images and tested on 4, with equal proportions of tumorous and non-tumourous samples occuring. After training the neural network, a 2% error was reached using the neural network to diagnosis the tumorous state of the input image.

4.7 App and Website

Both an app and a website were created to make the diagnosis more readily available to the general populace. In order to make the diagnosis applicable for more individuals, a cloud0based service was implemented through ASP.NET. In addition to the website, creating an app made the project even more applicable because of the amount of technology existing in society today. Just by going onto their phones or onto the Web, people are able to access these useful tools. Simple numerical inputs or image inputs are easily entered into either the app or website, making both more usable by the public. In addition, both the app and the website include all three diagnostic tests, which have different inputs based on the three datasets that the neural network was trained on. Moreover, to make the project reproducible, an open source repository was created through GitHub.

5 Conclusion

Numerous successful neural networks were created employing the accelerative learning algorithm. The final total error calculated for the neural network is 3.2% for the Wisconsin Original Breast Cancer dataset. However, one must consider that the error in the original dataset was 5% indicating that the network was able to adapt to the error, yet not to 100% accuracy. Despite the possibility of imprecision with the network, an algorithmic model would be extremely difficult and time consuming to create, thus the approximation allows for an efficient solution. The random initial weight values created difficulties in the code processing, as repeated generations of networks varied significantly. For the future, a heuristic determination of the starting weights would be more desirable in creating the network. The gradient method used to determine the values of the weights is not very accurate because it locates only local minimum as the threshold error value is set at .44%, where instead finding the global error minimum would be more accurate. However, the problem with finding global minimum is that the network might over fit the data, causing it to only recognize the training dataset and lose its adaptive nature to recognize other potential data points, rendering it useless to model complex functions. Furthermore, the error for the Diagnostic Wisconsin Breast Cancer dataset was even lower, at two percent for the testing error.

Although establishing potential for inaccuracy, the step function is an extremely important part of the data processing to recognize potential outliers. By weighting the output significantly, the network also becomes more accurate. Because the field, to which the network is being applied, allows no room for error, minimizing the amount of error received is essential. Thus the step function favored a false positive to ensure that any given diagnosis doesnt result in an unnecessary false-negative.

Finally, the 10 networks differed greatly, caused by the random initialized weight values. Even though the values are centered near zero for the range, they are still randomly placed according to the Gaussian distribution, causing a variance in convergence. This demonstrates the importance of weights on the network and how different random weights are able to lead to different solutions, indicating the self-structuring nature of a neural network.

The different variables that affect the network such as momentum, accelerative learning rate, and network size were demonstrated to have large effects on the error produced by the network. It was also demonstrated that these variables are very codependent on one another as indicated by the momentum, accelerative learning rate, and covariance experimental tests.

The image recognition portion of the diagnostic tests was successful, with an extremely low error of 2% being reached. However, there were several problems involved. The preprocessing was extremely situation dependent because each input image taken from the mini-MIAS database was of different sizes and made equal using black space on the sides. In addition, the neural network was trained and tested on an extremely small dataset. Increasing the dataset would not only make the experiment more reliable but also would give a more reliable error rate. It would also make the network more flexible to different images and types of tumors rather than being focused on the given training dataset.

The network also has many potential uses. It is an extremely flexible artificial intelligence structure that was able to adapt to three distinct datasets.It is not only applicable to breast cancer but any dataset that is or isnt able to be modeled by conventional method. The project allowed an application and website to utilize a generated neural network to quickly and accurately diagnose cases of potential breast cancer. These tools can not only be available and used by the general population, but also used in the medical field to alleviate the workload on medical professionals while they diagnose breast cancer.

6 Bibliography

Error Backpropagation. IMGUR. Web. <http://i.stack.imgur.com/pOR6t.png>

Neural Network. Karl Branting . Web. <http://www.karlbranting.net/papers/plummer/Paper_7_12_00_files/image016.jpg>

Dr. William H. Wolberg. Wisconsin Diagnostic Breast Cancer. Original. University of Wisconsin, July 1992. WEB. January 2014. <http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Original)>

Dr. William H. Wolberg. Wisconsin Diagnostic Breast Cancer. Diagnostic. University of Wisconsin, November 1995. WEB. January 2014. <https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)>

J Suckling et al (1994): The Mammographic Image Analyssi Society Digital Mammogram Database Exerpta Medica. International Congress Series 1069 pp375-378. WEB. <http://peipa.essex.ac.uk/info/mias.html>

Prof. Bebis. Wavelets (Chapter 7). University of Nevada. WEB. March 2014. <http://webcache.googleusercontent.com/search?q=cache:9N6XniIOuYAJ:www.cse.unr.edu/~bebis/CS474/Lectures/Wavelets.ppt+&cd=3&hl=en&ct=clnk&gl=us>

Stanford Medicine. Cancer Diagnosis. Name Stanford University. WEB. February 2014. <http://cancer.stanford.edu/information/cancerDiagnosis/>

Prof. Leslie Smith. An Introduction to Neural Networks. Department of Computing and Mathematics University of Stirling, April 2003. WEB. December 2014. <http://www.cs.stir.ac.uk/~lss/NNIntro/InvSlides.html>

Christos Stergiou and Dimitrios Siganos . Neural Networks. Imperial College London. WEB. January 2014. <http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html>

Aleksander, I. and Morton, H. An introduction toneural computing. 2nd edition Internat: Thomson Computer Press 1995. Print.

Fabien Torre. Datasets. Lille University, France, 1995. WEB. January 2014. <http://www.grappa.univ-lille3.fr/~torre/Recherche/Experiments/Datasets/#breast-cancer>

MD Anderson Cancer Center. Diagnostics Tests. University of Texas, WEB. March 2014. <http://www.mdanderson.org/patient-and-cancer-information/cancer-information/cancer-topics/detection-and-diagnosis/diagnostic-tests/index.html>

Error Backpropagation. Williamette University. WEB. January 2014. <http://www.willamette.edu/~gorr/classes/cs449/backprop.html>

William H. Wolberg and O.L. Mangasarian: "Multisurface method of pattern separation for medical diagnosis applied to breast cytology", Proceedings of the National Academy of Sciences, U.S.A., Volume 87, December 1990, pp 9193-9196.

Armando Bazzani, Alessandro Bevilacqua, Dante Bollini, Renato

Campanini, Nico Lanconelli, Alessandro Riccardi, Davide Romani. Automatic detection of clustered microcalcifications using a combined method and an SVM classifier. Department of Physics, University of Bologna, Italy Department of Electronics, Computer Science and Systems, University of Bologna, Italy National Institute for Nuclear Physics, Bologna, Italy.

Bache, K. & Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

Dheeba.J, Wiselin Jiji.G. 2010. Detection of Microcalcification Clusters in Mammograms Using Neural Networks. International Journal of Advanced Science and Technology. Vol. 19.

Steve Lawrence, C. Lee Giles, Ah Chung Tsoi. 1996. What Size Neural Network Gives Optimal ? Convergence Properties of Backpropagation. NEC Research Institute, 4 Independence Way, Princeton, NJ 08540 Department of Electrical and Computer Engineering University of Queensland, St. Lucia 4072, Australia.

U. Bottigli, P. Cerello, P. Delogu, M.E. Fantacci, F. Fauci, G. Forni, B. Golosio, P.L. Indovina, A. Lauria, E. Lopez Torres, R. Magro, G.L. Masala, P. Oliva, R. Palmiero, G. Raso, A. Retico, A. Stefanini, S. Stumbo, S. Tangaro. 2003. A Computer Aided Detection system for mammographic images implemented on a GRID infrastructure. Dipartimento di Fisica e Tecnologie Relative dell Universit di Palermo and Sezione INFN di Catania, Italy.

Chun−Lin, Liu. February 23, 2010. A Tutorial of Wavelet Transform.

CS. The Discrete Wavelet Transform. University of Toronto. WEB. February 2014.<http://www.cs.toronto.edu/~kyros/courses/320/Lectures.2013s/lecture.2013s.10.pdf>

7 Work Log

[12/27/2013]

- Initial commit of project solution and other important base files
- Researched the problem set. Researched error backpropagation and feed forward neural networks and researched comparisons among sigmoid activation functions.
- Added sigmoid activation class where each activation class is described by lambda definition of its mathematical form and the derivative thereof. Logistic, Hyperbolic Tangent, and Linear are defined in the neural network class.
- Moved the sigmoids into their respective classes
- Implemented abstract neurons
- Bias Neuron, Hidden Neuron, Input Neuron, Output Neuron implementations
- Cleaned the solution recursively and forced uniform white spacing.
- Fundamentally changing the model through which Neurons calculate values. Neurons now have void UpdateOutput(...) and void UpdateError(...) which handle those parameters internally instead of Get[Error/Output](...). The getters were then replaced by properties with protected setters and public getters. All changes occurred subsequently to subclasses.
- Added a random generator to create random numbers pertinent to the neural network
- Added the Connection class. The Connection class contains weight, posterior neuron, and anterior neuron properties. The update weight rule for individual connections was also implemented in double UpdateWeight()
- Added FeedForward() function to Connection.cs. It feeds the product of output from the anterior neuron and the weight of the connection forward to the anterior neuron.
- Formatted the BiasNeuron.cs
- Completed Layer-Neuron and Layer-Connection initialization
- Implemented the feed forward functionality of the network
- Cleaning the solution

[12/28/13]

- Implemented Error Backpropagation
- Calculates global error
- Adjusts output neuron to calculate proper error
- Implemented full learning rate control
- Implemented updating weights using learning rate

- Taught the network XOR
- Created a Gaussian distribution class, weights are set with a normal distribution between -1 and 1.
- Added net reset to Bias Neurons
- Added correct sum squared error calculation to the Network class.
- Switched difference terms in output error calculation
- Implemented Gaussian switch and changing weight sign
- Added momentum to weight training
- Added train method to Network Class
- Changed the constructor of the Network class
- Fixed a bug where neurons were not being reset
- Added None to Sigmoid for the input layer
- Finished base implementation, XNOR training and testing, with working bias neurons
- Cleaned the solution

[1/12/2014]

- Created Neural Network Library
- Cleaned the solution

[1/16/2014]

- Finished the DataPoint abstract class for training and neural analysis

[1/20/2014]

- Final trainer implementation with successful XOR network (2,4,1) added to the solution
- Removing debug parameter to train making it solution dependent

[1/21/2014]

- Added nudging functionality to the neural network

[2/1/2014]

- Added cancer dataset
- Fixed XOR program
- Fixed logistic curve of the trainer
- Cleaned the solution

[2/5/2014]

- Added dataset shuffling
- Added training error history with more accurate implementation and less processing power
- Added hyperbolic step function to sigmoids
- Moved nudging threshold down
- Added new dataset
- Fixed a fatal flaw with error backpropagation algorithm which made the convergence working
- Added RPROP connection and implemented initial RPROP, but discontinued RPROP algorithm
- Fixed the solution file
- Cleaned the solution
- Removed the readkey

[2/8/2014]

- Added saving and loading weights

[2/9/2014]

- Initial App project started
- Data calculations added

[2/11/2014]

- Final commit of the app
- Committing initial application for use with cancer diagnostics
- Merging conflicts on several branches

[2/15/2014]

- Removed conflict
- Cleaned the repo to account for normalized master networks
- Removed the cancer dataset
- Constructed XOR for working dataset
- Removed Wisconsin from solution
- Reduced and then increased the network size

[2/16/2014]

- Cleaning the project and leaving it in its most basic form
- Initial commit of clean BPNN
- Committing basic folder structure
- Establishing further directory structure with more changes
- Created initial commit of the numerical experiment
- Added an experiment class for running multiple tests
- Added first learning rate experiment
- Cleaned the solution

[2/18/2014]

- Worked more on the learning rate experiment and moved the dataset into the experiment folder
- Dataset and separation into training and testing dataset for state
- Created CancerData.cs to separate datasets and practicing thereof through Linq

[2/20/2014]

- Finished experimental analysis
- Created a control experiment
- Finalized learning rate experiment

[2/21/2014]

- Created new experiments for different parameters
- Researched about the input data information of the datasets

[2/22/2014]

- Fixed the hidden layer code
- Implemented nudging with standard deviation
- Add project experiment files into neural network class
- Fixed problems in experimentation code
- Added automatic stopping of training when nudging should usually occur
- More additions for proper analysis of error
- Optimized the network
- Worked on experimentation
- Added imaging project

- Committing the CS project
- Added new constants into the network
- Step Experiment
- Made 10 base networks for experimentation
- Further research and addition of comments
- Updated learning rate experiment
- Cleaned the output and solution

[2/23/2014]

- Added low pass experimentation
- Did the smallest experiment
- Conducted momentum experiment

[2/28/2014]

- Added conclusion experiment

[3/1/2014]

- Worked on the conclusion experiment
- Removed unnecessary output data
- Added website to the solution

[3/2/2014]

- Added covariance experiment data
- Cleaned the history
- Finished Wavelet transform
- Researched Gabor Filters
- Finished conclusion and worked on the website

[3/4/2014]

- Did image Dataset preprocessing

[3/7/2014]

- Worked on submission forms
- Completed initial submission form for proportionality test
- Added the second FNA test (detailed)
- Worked on the Journal

[3/8/2014]

- Added a diagnostic choice page and renamed diagnosis page to P-FNA page
- Committed the conclusion and step experiment
- Reorganized images and fixed image preprocessing
- Finished D-FNA portion of app

[3/9/2014]

- Finished D-FNA testing for less accuracy
- Created image experiment and finished it
- Continued work on the journal
- Created Graphs
- Created the work log

8 Appendix

sdf

8.1 Figures

8.1.1 Images

sdf

sdf

sdf

sdf

sdf

sdf

sdf

8.1.2 Graphs

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

8.2 Documentation

asd

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

8.3 Source Code

8.3.1 Neural Library

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

8.3.2 Experiment

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

8.3.3 Cloud Implementation

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

8.3.4 App Implementation

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

sdf

8.4 Data

8.4.1 P-FNA

sdf

sdf

sdf

sdf

sdf

sdf

sdf

8.4.2 D-FNA

sdf

sdf

sdf

sdf

sdf

8.4.3 MID